(2)

A RAND NOTE

Applying Artificial Intelligence Techniques to Strategic-Level Gaming and Simulation

Paul K. Davis

March 1988

40 Years 1948-1988
RAND

This publication was supported by The RAND Corporation as part of its program of public service.

Published by The RAND Corporation
1700 Main Street, P.O. Box 2138, Santa Monica, CA 90406-2138

A RAND NOTE

N-2752-RC

Applying Artificial Intelligence Techniques to Strategic–Level Gaming and Simulation

Paul K. Davis

March 1988

A research publication from
The RAND Strategy Assessment Center

40 Years 1948-1988
RAND

APPLYING ARTIFICIAL INTELLIGENCE TECHNIQUES TO STRATEGIC-LEVEL GAMING AND SIMULATION

Paul K. Davis

The Rand Corporation and the Rand Graduate Institute
1700 Main Street
Santa Monica, California 90406-2138
The United States

This paper describes a large-scale program melding rule-based modelling and traditional simulation in the problem domain of game-structured military strategic analysis. It then draws on the program's experience to discuss paradigms from artificial intelligence, concepts and techniques for representing knowledge in a policy domain having no body of acknowledged experts or experimental data, and lessons from managing the related research and software development. Finally, it discusses implications for the ability to reflect in policy analysis concepts of bounded rationality and organizational behavior.

1. INTRODUCTION

This book explores ways in which knowledge-based modelling can be integrated with traditional simulation modelling to produce something transcending the separate activities. This paper describes an operational prototype system accomplishing precisely this kind of synthesis, and a related research effort that may well be unique in scope, complexity, and relevance to interdisciplinary policy analysis. The paper proceeds chronologically to provide a realistic image of how a successful development evolved and of the extent to which concepts and techniques from AI have proved useful in each phase of the work.

2. PHASE ONE: THE CONCEPT OF AUTOMATED WAR GAMING

2.1 The Problem Domain and Initial Concepts for an Approach

The work described here had its origins in a U.S. government request in 1979 for a new approach to strategic analysis that would combine features of human political-military war gaming and analytic modelling. The request reflected rejection of sole reliance on traditional system analysis and strategic simulation models with their focus on stereotyped scenarios, simple measures of effectiveness, and purely quantitative analysis. From war gaming was to come an enriched global view with strategic nuclear forces present in a context of military campaigns, political constraints, alliances, escalation and de-escalation, and imperfect command and control (Marshall, 1982; Davis and Winnefeld, 1983). However, it was recognized that war gaming itself was not a solution: War games are slow, expensive, manpower intensive, and narrow in scope. Working through a single war game, however interesting, does not justify drawing conclusions because results depend positively on the players and because cause-effect relationships can be quite unclear. The government hoped, then, for a change of approach that would somehow retain the game character but permit greater analytic rigor.

Reprinted from *Modelling and Simulation Methodology in the Artificial Intelligence Era*, M. S. Elzas, T. I. Ören, and B. P. Zeigler (eds.), Elsevier Science Publishers B. V. (North-Holland), 1986, pp. 315-338. Reprinted with permission.

Upon reviewing the government's request, Rand concluded that to gain control over the enormous number of variables in a war game it would be necessary to automate the war game as suggested in Figure 1. In this concept, which was deemed radical in 1980, Rand proposed a game-structured interactive simulation in which any of the human teams could be replaced by automated models called "agents." Thus, one could have a closed simulation with agent pitted against agent, a mixed simulation with one or more human teams, or a computer-assisted human war game (Graubard and Builder, 1980).

Even at this stage of thinking the concept involved a synthesis of knowledge-based modelling (the Red, Blue, and Scenario Agents of Figure 1) and more traditional simulation modelling (the Force Agent, which would keep the game clock, move and keep track of military forces, assess the results of battle, and generally cope with the consequences of decisions). The concept also assumed that the decisionmaking agents (Red, Blue, and Scenario Agents) would depend primarily upon qualitative heuristic rules rather than on optimizing algorithms.

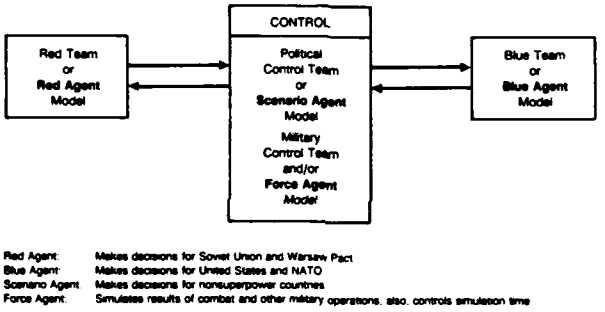

Figure 1
Schematic Representation of the Automated War Game

2.2 The Mark I System

Serious work began in 1980 when Rand was asked to build quickly a rudimentary demonstration system to illustrate its notion of automated war gaming. A conservative approach using existing capabilities at Rand was taken, but integration was complex. Some characteristics of the Mark I system were as follows:

- The Red Agent made decisions by (a) comparing a 15-dimensional characterization of the current world state with a set of world states previously considered and identified in data, (b) choosing the world state in data that was "closest" to the current one (as defined with a Euclidean metric), and (c) following If ... Then ... instructions associated with that of the state in data.

- The Scenario Agent was an experiment with rule-based modelling in an English-like Rand AI language called ROSIE (Fain, Hayes-Roth, Sowizral, and Waterman, 1982) that allowed users to review and change rules interactively.

- The Force Agent consisted of simple combat simulation models developed in previous Rand work.

- There was a Blue team rather than a Blue Agent, all interfaces were manual, and the system lashed together three computers and three programming languages.

The Mark I system successfully conveyed to government reviewers a sense for the project's vision and demonstrated not only that decisionmaking agents could be built, but also that automated war gaming could allow users to examine a wide range of scenarios. Indeed, it strongly encouraged focusing on the very scenario variables that are so often taken for granted in policy studies (Builder, 1983). The reviewers were also impressed with the experimental Scenario Agent (Dewar, Schwabe, and McNaugher, 1982), because they placed great value on being able to review and change the underlying rules of a model. This endorsement of transparency by senior and mid-level officials had a major impact on subsequent work.

In spite of their interest, the reviewers were by no means sanguine. They were skeptical about moving from a simple demonstration system to something with a high degree of military content. They feared that the Mark I system might be "another example of computer scientists making great promises on the basis of a 'toy problem'." Nonetheless, the Phase One effort was successful in gaining support for further work and laying an empirical base of experience for the research team that would prove important for the next five years.

3. PHASE TWO: CONCEPTS FOR IMPROVING THE SUBSTANTIVE CONTENT

3.1 Prelude

After a year's funding hiatus, work began again in earnest in early 1982 with the government stipulating that there was to be no additional computer activity until the concepts had been developed for greatly improving the military content of the approach.

Key decisions were made rather quickly. The Mark I pattern-matching approach to the Red and Blue Agents was inappropriate for advanced work because it was neither transparent nor readily scalable to more complex work. After a review of alternatives based on a survey of the AI literature (Steeb and Gillogly, 1983), it was decided early in 1982 that decision modelling would revolve around "analytic war plans," which would be new constructs (Davis, 1982) based loosely on the scripts of AI (see, for example, Schank and Abelson, 1977, and Carbonell, 1978). The rationale for this was that by using script techniques one could make early use of the substantial knowledge that exists about real-world military operations and plans. The alternative of developing a goal-directed model to create such complex plans within the simulation appeared (at least to the author) to be a prescription for delays and potential failure.

The script approach was itself no panacea and would have to be extended significantly to permit the Red and Blue Agents to change their analytic war plans at appropriate times and to orchestrate the change of plans intelligently. Although there is an insidious temptation to turn strategy problems into decision trees, which would make scripts easy to use, the reality is that one does not know even which events will occur, much less in what order or precisely when.

This implied the need for a building-block approach in which primitive plans in the form of scripts would be combined and used to fit the circumstances (Davis, 1982).

The model for Red and Blue Agents emerging from this thinking was to have three "levels." The National Command Level (NCL) would pick analytic war plans, which would be the scripts containing general instructions for Area Command Levels (ACLs), with different ACLs for different regions worldwide. The ACLs would refine the action instructions based on the current state of the simulation (e.g., they might choose one or another branch of an analytic war plan). The Tactical Command Level (TCL) submodels, which have subsequently been reinterpreted, would refine instructions to the level of tactics and would determine more precisely what the Force Agent would be asked to do with military forces. Thus, one could say that the NCL submodel would pick a script and the ACL and TCL would "fill in slots of that script." Precisely how this would be accomplished was unclear, although some of the workers envisioned something akin to AI chess-playing programs with large numbers of look-ahead projections searching for good stratagems within the general structure imposed by the analytic war plan (Steeb and Gillogly, 1983).

There were two other important concepts affecting the Red and Blue Agents: (1) because of fundamental uncertainties about superpower behavior, there would be alternative Red and Blue Agents referred to informally as "Sams" and "Ivans," and (2) as suggested above, the Red and Blue Agents would conduct look-aheads to test potential analytic war plans before making final decisions, look-aheads consisting of a game within the game using Red's image of Blue and vice versa (more on this below).

Another part of the early thinking in 1982 dealt with the characteristics desired for a future Force Agent. Several conclusions were especially important (Davis, 1982; see also Bennett, Bullock, Jones, and Davis, 1985):

- The emphasis would be on breadth rather than depth, and well-intended suggestions to add ever-increasing detail would be resisted. The goal was to be the capability to answer many "What if?" questions quickly, with a strategic rather than tactical perspective. This implied aggregation and parameterization.
- The simulation would be self-contained with no special effort to insert other large-scale models as modules (something likely to increase technical problems and confusion).
- To the extent permitted by finite imaginations, the attempt would be made to reflect all phenomena having an important bearing on the strategic-level outcome--even if the phenomena were difficult to quantify or simulate. As a matter of principle, "scripted models" would be used, which would draw upon separate studies, judgments, or prudent speculation, rather than omit important phenomena altogether (Davis, 1982).

A scripted model might be as simple as a provision allowing one of the superpowers to create a crisis in region x as the result of events in region y, with the precise nature of the crisis left unspecified. In this case, the model would merely post a symbol in the world state: "Crisis exists in region x." In other cases, a scripted model might be quantitative but simple. For example, it might decree that the military forces of Side A would advance through a particular region on a rigid timescale based on a separate study.

There was nothing novel about the individual examples of scripted models, but establishing their widespread use as a matter of principle was unique. It also met resistance until the scripted models began proving their value by allowing the research team to address issues policymakers know are important but that modellers commonly ignore because of uncertainties or "softness." The scripted models are especially valuable in dealing with issues of escalation, where nations react to symbols (e.g., the fact of an invasion or the crossing of some other tacit boundary) more than to details, and in examining parametrically the potential significance of military events that are difficult to simulate in detail (e.g., events affecting command and control). In passing, it might be observed that resistance to treating qualitative ill-defined events in simulations with scripted models was analogous to the resistance received in the 1960s and 1970s by those who first questioned rational-analytic models (Simon, 1980) and those who insisted on including important but subjective cause-effect relations in policy simulations (Forrester, 1969).

3.2 The Mark II System

Although most computer work had been deferred, it was deemed essential to gain additional empirical experience. Thus, by the summer of 1982 manual walkthrough experiments were conducted to gain experience with analytic war plans, an improved Scenario Agent (Schwabe and Jamison, 1982), and the use of multiscenario experiments to derive policy conclusions. These experiments improved intraproject communications and morale and provided the basis for a coherent status report. Their value was analogous to that of the rapid prototyping frequently discussed in the literature.

Some of the principal lessons from this period of theorizing and experimenting were the following:

- Even simple analytic war plans could effectively communicate a sense for objectives, strategy, and interrelationships. Although the plans existed only in notebooks, it would clearly be possible to encode them.

- The notion of Red's Blues and Blue's Reds (i.e., each side's perceptions of the other) could reflect analytically something discussed by international relations specialists for years: that misunderstandings about the opponent can be the origin of unintended escalation and other forms of disaster (see Glaser and Davis, 1983, and classic references therein; see also the more general discussion of policy-relevant games by Elzas, 1984).

- As noted earlier, the script concept would have to be extended substantially to cope with plan changing and resynchronization (one of the general problems facing variable-structure simulation, as discussed by Zeigler elsewhere in this volume).

- Interdisciplinary work was greatly hindered by the jargon of AI and computer science more generally; domain-specific language was greatly to be preferred if misunderstandings were to be minimized and the vision sharpened. There were serious culture gaps among AI specialists, other computer scientists, and modellers/programmers concerned with military issues.

- The transparency of individual rules in English-like computer code was misleading: it was very difficult to review and

comprehend entire rule modules, in part because it was difficult to judge completeness and to see the logic in a compact physical space.

- o There was an insidious side to the practice of heuristic modelling--in taking the pendulum swing away from rigorous quantitative modelling, there was a tendency to write ad hoc rules without appropriate structure or theory. It was sometimes difficult to judge the quality of rule sets.

The Phase Two work led to a government decision authorizing construction of a prototype system over a two-year period (1983 and 1984), an effort that would require approximately twice the previous phases' efforts (approximately 70 man-years were expended between 1980 and the end of 1984).

4. PHASE THREE: DEVELOPMENT OF A PROTOTYPE SYSTEM

Phase Three began early in 1983 and continued intensively for two years (and, in a sense, for 2-1/2 years, since consolidation of progress required at least six months in 1985). The result was a prototype system for automated and interactive war gaming. This section describes the design and implementation of that prototype, again with a view toward indicating realistically how progress was made. At present (late 1985), work is under way to develop and transfer to the government a first-generation operational version of the Rand Strategy Assessment System (RSAS). That work will continue through 1986.

4.1 Choosing a Programming Language

One of the earliest decisions in 1983 was to choose a programming language. The requirements were: (1) speed (less than a half day for an automated war game processing thousands of rules and large numbers of numerical algorithms with programs having tens of thousands of lines of code); (2) transparency of decision rules and decision processes (explanation facilities); and (3) transportability of the eventual system for automated war gaming. The decision was to work with two programming languages, one for the Force Agent simulation, the other for the rule-based decision models. The former would be the existing C language; the latter would be new and would require considerable development. What follows summarizes the principal considerations (Shapiro, Hall, Anderson, and LaCasse, 1985a):

SIGNIFICANT FEATURES OF LANGUAGES USED

C Language and UNIX Operating System (major characteristics)

- o portable among modern computer systems
- o available and familiar at Rand
- o efficient
- o powerful and flexible (e.g., C/UNIX allows
  hierarchical files, complex data structures
  accessed by strongly-typed pointers,
  long variable names improving readability of
  self-documenting code, and structured programming)
- o suitable for building a fast readable preprocessor
  language for the decision models (using UNIX
  utilities such as LEX and YACC)

RAND-ABEL™ Programming Language (initial requirements)

- o fast (execution speeds of seconds for decision models)
- o readable and modifiable by nonprogrammers
- o easily extended
- o identify errors early (via strong typing)

Adopting C/UNIX raised difficulties for the Force Agent programmers familiar with languages such as Fortran and PL/1, but they came rather quickly to like C/UNIX, which is not usually mentioned in connection with simulation modelling. The basic reason for their enthusiasm was that the simulation has many features akin to those of operating systems (e.g, extensive use of pointers to a centralized database necessitated by interactions among parts of the simulation, interactions of special interest for strategic-level global war gaming). C/UNIX is a strong language for such applications.

The history and rationale for the RAND-ABEL language are described in Shapiro et al (1985a). Two factors were especially important. The first was the speed requirement, which ruled out Rand's ROSIE language, even though ROSIE has been successfully used on many other AI applications successfully. The second factor was more subtle: it was observed that early decision models such as the Scenario Agent did not rely upon generalized inference with AI-style "search," although the ROSIE language in which it was written had such capabilities; this, and the impression that the same would be true for more advanced models, suggested that the new programming language could be highly procedural and would not have to possess a general inference engine. Thus, RAND-ABEL was originally conceived as a fairly simple pre-processor for the C language. In reality, it has developed into a sophisticated language with some unique and attractive features (especially tables) that will be discussed later in the paper. It has indeed proven very fast (only three times slower than C itself), resulting in execution times less than 1 millisecond per rule on a VAX 11-780.

4.2 Model Architecture and System Design: Managing Complexity

The next issue was to tighten the conceptual model and develop something approximating system specifications. Because concepts were still evolving rapidly and because those concepts were often abstract because of their novelty, it was infeasible to develop rigorous system specifications and a conscious decision was made to forego attempting to do so except in specific problem areas. Indeed, the reality is that many system specifications were not developed until 1985--after the prototype system had been successfully constructed and demonstrated.

What did exist in 1983 was a conceptual model designed with a strategic-level perspective and a rather high-level system overview (Hall, Shapiro, and Shukiar, 1985). Figure 2 describes the conceptual model as conceived early in 1983 (Davis and Stan, 1984).

The first column of Figure 2 describes the top level of the game: Red, Blue, Scenario, and Force Agents "take turns," although the order and time spacing of moves vary with events in the simulation and moves by one agent are based on only imperfect knowledge of other agents' prior moves because of time delays and command and control limitations. Recall that the Force Agent is conducting the simulation and keeping the simulation's clock; the role of the other agents is to make decisions about the actions to be simulated by Force Agent.

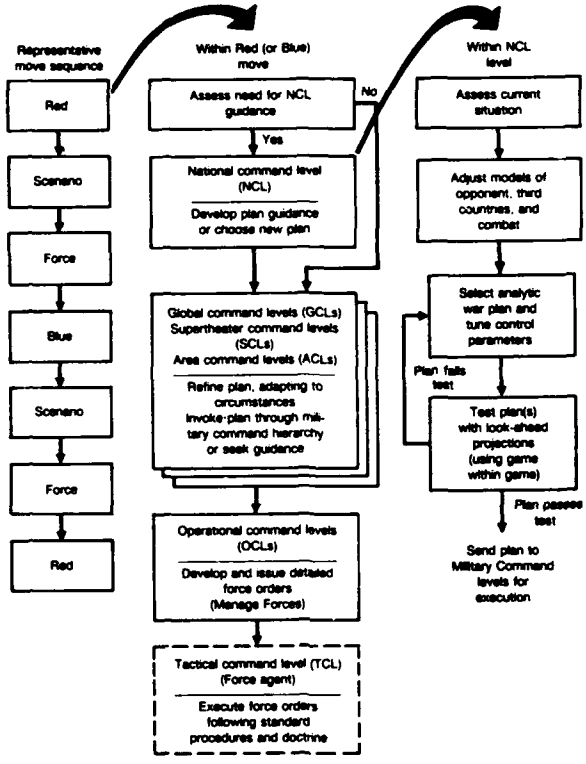

Figure 2
Conceptual Model of the RSAC Simulation

Using Red for the sake of example, the second column shows that within a given Red move the first issue is whether Red is satisfied with results using its current analytic war plan. If so, Red merely continues with that plan--although it may be necessary to refine the plan or to decide which of several plan branches to take. The refinement and execution of a plan is accomplished by modules identified with military command levels.

If the Red Agent is not satisfied with its current analytic war plan--as determined by tests contained within that plan (e.g., tests of whether Blue is escalating or successfully defending)--then Red's NCL must have the opportunity to change that plan. The process for doing that is suggested in the third column (and discussed at more length later in the paper). The character of the NCL decision process is not that of a script, but rather something more nearly like rational-analytic thinking--but with mostly heuristic decision rules.

The third column of Figure 2 indicates that after the NCL has tentatively chosen a new plan it can test that plan by conducting a look-ahead--a game within a game. However, in doing so it must make

assumptions about its opponent, third countries, and the laws of war--all of which may be wrong. Just as there are alternative Red and Blue Agents (the Sams and Ivans), so also can there be alternative Red's Blues and Blue's Reds. In principle, this perception-related recursion could extend to infinite depth, but actual practice avoids look-aheads within look-aheads: Agents within a look-ahead can project their opponent's behavior using current information and heuristics or simple algorithms, but they do not call upon the full game machinery for a game within a game within a game. Realistically, this seems to sacrifice very little credible information.

Figure 3 elaborates on the internal structure of the Red and Blue Agents, emphasizing a more complex hierarchical character than originally conceived. The Global Command Level (GCL) is an abstraction combining, for example, the Blue functions of the Joint Chiefs of Staff, State Department, and White House Staff. The Supertheater Command Level corresponds to something like NATO's Supreme Allied Commander in Europe (SACEUR) and, simultaneously, the U.S. Commander in Chief in Europe. The Area Command Levels, then, correspond to individual theater-level commands (e.g., the NATO commander for Europe's Central Region). The number of subordinate commands (i.e., SCLs and ACLs) can be changed easily, as can the physical regions over which they have authority. The Operational Command Level (OCL) models are really service models for managing forces. One function of an OCL model might be to decide on a daily basis what fraction of a given theater's air forces should be used for interdiction bombing rather than air defense. These models rely upon a mixture of quantitative algorithms and qualitative heuristics and are responsible for reasonably fine-grained decisions about operations.

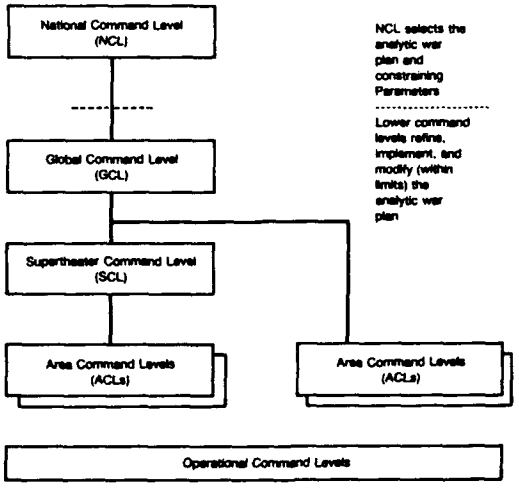

Figure 3
Hierarchical Structure of Red and Blue Agents

The outputs of the GCL, SCL, ACL, and OCL models are the force orders to which the Force Agent responds when it next conducts the simulation. In executing force orders, the Force Agent can apply different tactical logic for Red and Blue to reflect asymmetries

between Red and Blue military forces, doctrine, and capability. Thus, part of the logic defining the abstractions known as the Red and Blue Agents is actually located within the Force Agent rather than the decision models but the decision models can guide the Force Agent's logic by setting appropriate parameters. For example, an ACL may specify a forward defense rather than a fallback to some defense line in the rear and an OCL may specify certain constraints on the use of different national forces within an alliance; the Force Agent will then perform regular reallocations of ground forces among combat axes consistent with the guidance received.

Figure 4 is an influence diagram showing that the various command levels communicate through a rigid line of authority. By writing appropriate rules, however, the real-world phenomenon of a national leader communicating directly with a tactical commander can be simulated: the mechanism for doing so amounts to having the intermediate command models pass on the NCL's instructions without delay or adjustment. The design also specifies that (a) diplomatic exchanges occur between GCLs and (b) the NCL uses information resulting from Force Agent and Scenario Agent activities, but does not send instructions back, communicating instead with its lower command levels.

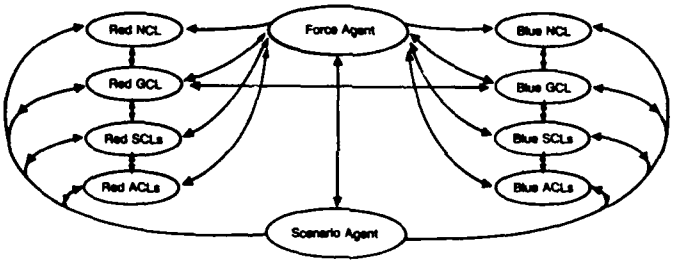

Figure 4
Influence Diagram for RSAS

Figure 5 shows an aggregated data-flow diagram of the system as it existed in late 1985 (Davis, Bankes, and Kahan, 1985). The system is highly centralized because of the many interrelationships that exist among types of decision and types of military actions worldwide. Thus, there are many global data variables for both the decisionmaking agents and Force Agent. Access to information is controlled through a data dictionary. Humans can interact through a data editor or a currently separate Force Agent interface.

It is impractical to show a control-flow diagram for the system because the control flow varies drastically from run to run. Figure 6 shows the system's subprograms (or, loosely, "objects"), with System Monitor represented as the Main Program determining control flow. It is System Monitor, for example, that contains the "wake-up rules" determining when a particular agent should have a move, and the order of moves among agents when several wish to move at the same time. The mechanism for wake-up rules is fairly complex. For example, if a Red Area Command Level model decides on a course of action and issues a series of orders, it will also specify the conditions under which it will want to move again (e.g., every day, or if any of certain events occur). These conditions will be reflected by data items in the World

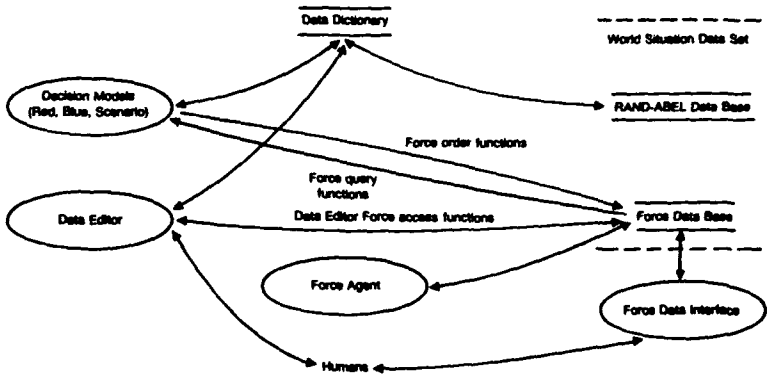

Figure 5
Data Flow Diagram for RSAS

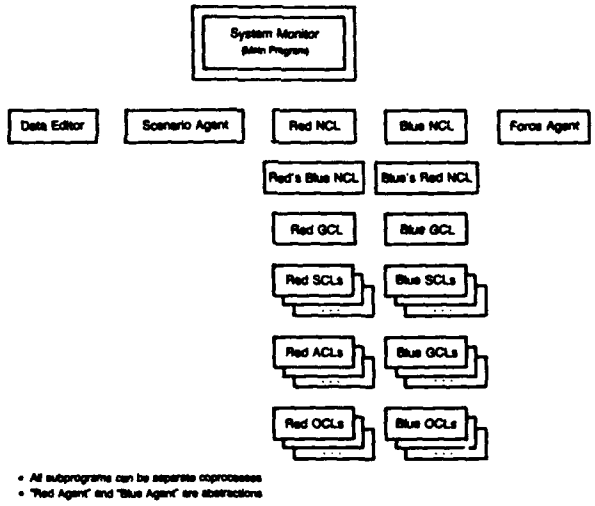

Figure 6
Top-level Subprograms (Objects) of the RSAS

Situation Data Set (WSDS). As the Force Agent conducts the simulation, it tests the conditions on a regular basis; if one of them is met, it sets another data value indicating that one or more particular agents want to awaken and move. It is then System Monitor that takes control and decides which Agent is to be awakened next.

It was recognized early in 1983 that implementation of analytic war plans with a script-like character would be greatly simplified by using co-routines, so a co-routine capability was provided in RAND-ABEL by Edward Hall for the UNIX environment (Shapiro et al.,

1985a,b). By late 1983, it was decided to use hierarchical co-routines structured to correspond reasonably well to command and control relationships in the real world (i.e., relations of the sort suggested in Figure 3).

One reason for including the design diagrams is to emphasize the difference between the system discussed here and more typical expert systems. Those differences include: (1) complexity (the system discussed here can be considered to include many separate expert systems as well as programs of a rather different character); (2) the requirement for hierarchical structuring; and (3) the requirement for difficult system programming rather than application of generic expert-system software.

It may also be worth noting that the system is definitely "object oriented" intellectually. The various agents represent a nearly decomposable hierarchy (Simon, 1980) of natural entities that operate independently to a first approximation but interact significantly in controlled ways (e.g., by engaging in crisis or conflict). They also communicate in the normal sense of that term (e.g, through diplomatic messages). On the other hand, RAND-ABEL does not have some of the features of object-oriented languages such as SMALLTALK (Goldberg and Kay, 1976) or Rand's ROSS (McArthur and Klahr, 1982). In particular, the mechanisms for message passing and other interactions are quite different.

4.3 Representation of Knowledge: General Comments

The design of the Rand Strategy Assessment System (RSAS) had to address many basic issues, including: (1) what entities should be focused upon in the game-structured simulation, (2) when should each entity move, (3) how should the various entities communicate, and (4) how should the various entities decide what to do in a move (decision style). The answers required domain-specific knowledge if the resulting simulation was to be comprehensible and achieve a level of phenomenological verisimilitude (or, at least, a naturalness and plausibility).

A sometimes implicit consideration in system design, but one about which the author was especially concerned, related to the absence of experts and empirical data. There are national security specialists (including the author), but there are no "experts" in the sense that term is used in AI. And, fortunately, there have been few superpower crises in the last forty years, and none of them has brought the world into general war. One consequence of this is that a system attempting to simulate a variety of potential crises and conflicts had to achieve higher standards of transparent logic than would, for example, an expert system that could demonstrate its validity empirically. Also, if the system were to be useful in understanding issues of deterrence, crisis stability, and military strategy more generally, the various rule-based agents would have to exhibit human-like behavior and thought processes. At the same time, model development would have to be highly analytic: For this application no expert was available to provide all the rules from his intuition, and yet it was essential that care be taken to assure some level of completeness--holes in the knowledge base could be much more serious here than in a typical expert system.

These factors had many consequences, among them the use of substantial domain-specific knowledge in every facet of system design. By contrast with expert-system models in which it has proved useful to

separate discrete entities such as a scheduler, inference engine, and knowledge base (see, for example, Hayes-Roth, Waterman, and Lenat, 1983), in the present work such an approach was neither feasible nor desirable. To the contrary, knowledge has been reflected in many different places and in a variety of different representations. Furthermore, the logic of RSAS models makes only minimal use of "general search" or generic inference techniques. Instead, the control logic tends to be highly structured (procedural)--something consistent with many classic AI efforts (e.g., those involving decisionmaking theory, organizational theory, or script), but quite different in character from much of the current expert-system work on goal-directed search with programming languages possessing inference capabilities using generalized search.

4.4 Knowledge Representation in System Architecture

The RSAS system design, then, employs substantial "meta knowledge" in the sense of Hayes-Roth et al (1983). This meta knowledge about the domain (military strategic analysis) affected both system architecture (e.g., the objects of attention) and the system's top-level control structure.

SYSTEM CHARACTERISTICS REFLECTING DOMAIN-SPECIFIC META KNOWLEDGE

- Number of high-level players in game (Red and Blue)
- Command-control hierarchy for players (see Figures 3,4)
- Order of moves
- Top-level process model for Red and Blue Agent moves (see Figure 2)
- Process model for Scenario Agent moves (Perception-Response)

Meta knowledge also determined the characters of the various Red and Blue submodels (which will be discussed further below):

MODELS AND MODEL CHARACTER

National Command Level	Rational, analytic, global in perspective, and not bound by history: Strategic Behavior
Military Command Levels (GCL, SCL, ACL)	Follows a prescribed plan, albeit with key decision points and adaptations, using many building-block procedures that are individually complex but practiced: Organizational (Cybernetic) Behavior
OCL	Manages forces on a routine basis using a combination of algorithms and heuristics (Service Behavior)

Given the variety of models and the complexity of the simulation, there is no single "knowledge base" in the RSAS, nor a single way to change knowledge. Conceptually, however, one can still talk about the knowledge base and how to change it. The options available are summarized below, with the number of pluses indicating how easily the change can be made.

WAYS TO CHANGE KNOWLEDGE IN THE RSAS SIMULATION

	NCL	GCL, SCL, ACL	OCL	Scenario Agent	Force Agent
Substitute entire rule sets (e.g., switch Sams, Ivans, or the scripts available to Red and Blue Agents)	+++	+++			
Change individual rules or algorithms	++	++	++	++	+
Change parameters within rules or algorithms	+++	+++	+++	+++	+++
Change parameters characterizing forces, capabilities, laws of war, etc.					+++

This listing also indicates the RSAS' hierarchical approach to model modification. All of the models are constructed so that usually one can implement changes (e.g, for sensitivity tests) by merely changing a parameter--either in data beforehand, or interactively during a simulation. In addition, however, the rules written in RAND-ABEL can be read and changed so easily that they can also be considered, in a sense, part of "data." *By the end of 1985 RAND-ABEL will also have* interpretive features making recompilation unnecessary after most changes of decision-model rules. Currently, a reasonably typical recompilation can take 15 minutes.

4.5 Knowledge Representation Within Analytic War Plans

As mentioned earlier, the GCL, SCL, and ACL models are supposed to demonstrate organizational behavior. Scripts are especially suitable here because they can capture detailed procedural knowledge (i.e., knowledge about how to do things, which may require dozens or hundreds of individual instructions). The scripts of theatrical plays are a good metaphor for military war plans because both proceed from start to finish without backtracking, and both have natural phases (e.g., one must alert forces before they deploy, and one must deploy them before they can enter combat). Just as an actor knows what to say or do next because he already understands where he is, so also is it easy to write military-command-level decisions within a coherent analytic war plan: the script format reflects implicitly a great deal of context that otherwise would have to be explicit. In effect, if the plan is applicable, then various actions go together and certain decision points and adaptations are predictable and to some extent schedulable. The participant in the plan need not worry about the larger context.

The analytic war plan describing the decision logic for a particular ACL commander is organized hierarchically by phases, within phases by moves, and within moves by technical complexity. For example, the first phase of a given plan might involve placing forces on alert; this might occur over a series of moves because of political factors and the need to call forces up from reserve status. Within a move that, for example, alerted military airlift, there would be a number of specific orders understandable by the Force Agent, or by an OCL model that would in turn issue detailed orders understandable by the Force Agent.

Although simple scripts might consist of nothing but a sequence of instructions, analytic war plans must also possess decision rules determining (as a function of the world state): when to proceed from one move or phase to another; whether to take one or another branch of the plan; what type of guidance should be given to OCL models managing forces; whether to request something from higher authority (e.g., more forces or authorization for some action); and whether to notify higher authority that the plan is no longer working as intended and that higher authority may wish to modify the plan or change it altogether.

Since the analytic war plans are to be building blocks for reasons discussed earlier in the paper, they must also contain logic allowing them to adapt to circumstances. That is, when a given plan (script) is first invoked, it must establish its own context. As a minimum, it must determine what the previous plan has already accomplished (Are the forces already alerted? Have they been deployed?). In principle, the plan might have to undo some of what the previous plan had ordered (e.g., disengage forces prior to starting a separate offensive).

In the general case, such resynchronization would be an extremely difficult problem in variable-structure simulations such as this. However, in practice it has proven possible to proceed with relatively simple domain-specific expedients. The principal technique is to build analytic war plans in families, with all plans of a family having a common understanding of plan phases so that if one plan replaces another it can test to see where it should begin. Using only a few indicators of context, it has been possible to develop a fairly large number of building-block scripts for the Red and Blue Agent prototypes.

4.6 *Knowledge Representation In National Command Level Models*

Clearly, knowledge representation for the National Command Levels must be altogether different from that of the script-based models if the NCLs are to exhibit "strategic" behavior. The most important issues in representing knowledge are probably (Davis and Stan, 1984; Davis, Bankes, and Kahan, 1985): (1) decision style, (2) strategic framework for organizing information, and (3) characterization of alternative NCL models (the various Sams and Ivans alluded to earlier).

The issue of decision style is especially interesting because the choices range from decision-analytic styles, with decision trees and utility calculations, to various styles associated with cybernetics or bureaucratic politics (Simon, 1980, 1982; and Steinbruner, 1974). Consistent with the desire for phenomenological fidelity and transparency, the approach adopted is akin to the idealization of what many real-world decisionmakers attempt (see, for example, Janis and Mann, 1977). Figure 7 illustrates this idealized decisionmaker as a process in which the decisionmaker first decides whether to decide, then examines a broad range of options (taking pains to assure that he receives a good set of choices to consider), contemplates the options with a variety of objectives and values in mind, searches for tie-breaking information, assimilates that information, reviews, and decides. Also, the decision model includes plans for implementation, contingencies, feedbacks, and control.

The NCL model is described in Figure 8. Note that the NCL model makes a situation assessment that includes "learning" as well as "projecting." It then specifies escalation guidance, operational objectives, operational strategy, and detailed controls within more generic war plans. Each such decision limits further the number of

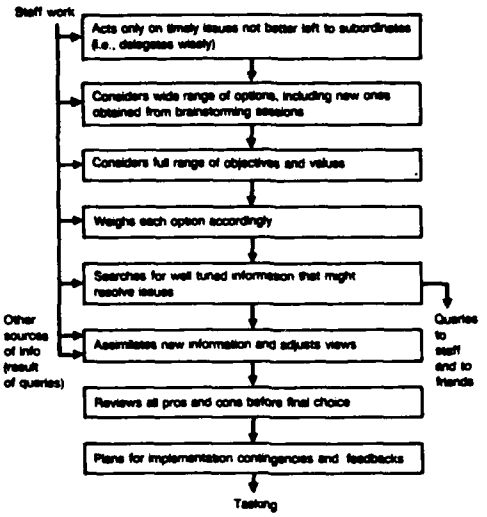

Figure 7
Actions of a Mythical Ideal Decisionmaker

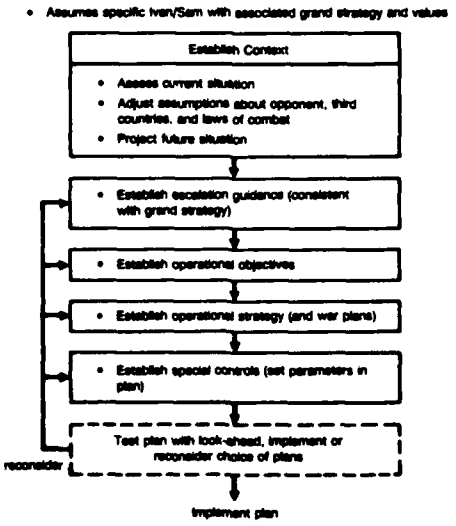

Figure 8
Decision Process Within NCL

available plans to be considered until a single plan is selected. The NCL then performs a look-ahead projection to see whether the tentative plan is likely to succeed; if so, it proceeds (or compares results for two or three plans); if it is unsuccessful, it reconsiders (with feedback logic determined by rule-based preference orders for plan testing and the requirement that alternative controls be tested first, then strategies, then objectives, then escalation guidances).

The relationship between the NCL model and the idealized decisionmaker of Figure 7 is that both describe decisionmaking as a process, both rely primarily upon heuristic rules rather than optimization or general search algorithms, both recognize the potential for obtaining and using special information, and both involve trading off conflicting objectives and values. The NCL model makes explicit the process of situation assessment and has the options (plans) as inputs rather than producing the options itself. In summary, then, the overall decision styles are similar, but there are significant technical differences. Finally, it should be noted that the design permits treatment not only of very good decisionmaking but also decisionmaking degraded by limitations of information, perception, mindset, and style. This is essential if the models are to be a useful analytic mechanism for studying issues such as deterrence and crisis stability.

As mentioned earlier, the fundamental uncertainties about national behavior necessitate use of alternative Red and Blue NCL models. Before building a given model, one must form a strong image of the particular NCL--its personality, grand strategy, and "temperament." There is a variety of techniques for crystallizing concepts on these matters before beginning to write the decision rules that constitute an NCL model. For example, essays such as the following prove useful:

> Model X is somewhat aggressive, risk-taking, and contemptuous of the opponent, but is ambivalent--believing that the opponent can sometimes be aggressive and irrational. He believes his military doctrine is essentially correct, although not always applicable. He is strongly averse to nuclear war....

Such essays communicate a sense for a given model, but they are imprecise. More useful are some tree-like diagrams that one can work out on blackboards to suggest the decision points and predispositions a given type of national leadership might have (Davis, Bankes, and Kahan, 1985). Also useful is a methodology involving formal attributes within which one fills out a check list for the decision model, a check list characterizing the model's attitude toward use of nuclear weapons, its willingness to engage in risky tactics, and other factors. The purpose of the essays, grand-strategy trees, and some formal attribute lists is to clarify an image of the particular Red or Blue NCL before the rule-writer attempts to write rules. This type of procedure is essential for achieving coherence and logic.

Next to be considered is the question of how to organize information within a strategic framework. Three concepts have proved especially useful: conflict levels, conflict states, and a hierarchy of variables for situation assessment.

The concept of conflict level has become familiar in strategic analysis since the seminal work of Kahn and Schelling in the 1960s. One constructs an escalation ladder in which, for example, nuclear war is "higher" than conventional war. There can be a large number of rungs on the ladder if one wants to distinguish carefully between

levels and types of nuclear use, types of crisis, and so forth. The ladder concept can be quite useful, although there are serious ambiguities when one applies it to complex problems (e.g., is war in one theater higher or lower than war in another theater?). A more sophisticated concept is that of conflict state (Davis and Stan, 1984), where the state is described in several dimensions such as conflict level by theater and side, and the presence or nonpresence of certain symbolic actions, again by theater and side. Figure 9 shows a simplified version of such a conflict state in which dots indicate the types and locations of certain military activities within a large but constrained set of scenarios focused on Southwest Asia, Europe, the high seas, and the intercontinental arena. Although rather simple, Figure 9 summarizes much of what senior players tend to focus upon in pondering high-level abstractions in political-military war games.

	Theater of Warfare				
Conflict Level	SWA	Naval	Europe	Space	Intercontinental
General Strategic Nuclear					
Counterforce Strategic Nuclear					
Demonstrative Strategic Nuclear					
General Tactical Nuclear	•	•	•		
Demonstrative Tactical Nuclear					
Biological					
Chemical	•				
General Conventional	•	•	•	•	
Demonstrative Conventional					
Regional (one superpower)					
Crisis					

Figure 9
A Simplified Conflict State Space

The last concept to be mentioned here is that of hierarchical situation assessment. The objective in this concept is to have models capable of summing up a situation as follows, much as would a human decisionmaker:

> Well, gentlemen, if I understand what you've been telling me... then...our current situation is pretty good: we've achieved our principal objectives (though not everything we sought). We could push on, but prospects are marginal and risks high. Therefore,...

Note the emphasis, in this summary, on high-level abstractions such as situation, prospects, and risk. There is no mention here, for example, of the detailed force ratio on the third axis of advance in the second of three active theaters. However, the abstractions depend on layers of staff work that do reflect numbers and other details.

It follows, then, that situation assessment in an NCL model should depend primarily on a few high-level variables, each of which depends on a number of intermediate variables, which each depend on still more lower-level variables, and so on. An example here is that the assessment of Risks should depend on strategic warning, tactical warning, sensitivity of projections to assumptions, ultimatums

received from the opponent, and assessment of the opponent's intentions and will. These, in turn, would all depend on more detailed events in the simulation (the current state and history).

The current NCL prototype models contain about 4000 lines of RAND-ABEL code each, which is the equivalent of 10,000-20,000 lines of C code. The rules are expressed primarily in terms of RAND-ABEL decision tables more or less as follows, although the example is contrived.

```
If Current-situation is Eur-gen-conv
Then
  Decision Table
  Basic-status Other-status  Prospects Risks   / Decision
  ============ ============  ========= ======   ========.
   goals-met    good           good      low      terminate
   goals met    good           good    medium     terminate
   [... many other lines...]
   losses       bad            bad     high       surrender.

Else If Current-situation is Eur-demo-tac-nuc
Then ...
```

Thus, the rules are modularized first by Current-situation using a fairly complex escalation ladder to identify current-situation's values. Within those modules, the rules are hierarchically structured to depend primarily on high-level variables such as Risks, which are then assessed elsewhere in the program in terms of strategic warning, tactical warning, and other variables.

The RAND-ABEL decision table is to be read as follows. The first line of the table's body is a rule saying that "If Basic-status is goals-met, other-status is good, prospects are good, and Risks are low, then the NCL decision is to terminate." This table is executable RAND-ABEL code (not data) equivalent to a decision tree. The table format allows the reader quickly to "see the whole" and to assess completeness and flow. There exists an interactive table generator to assist analysts writing rules to specify format, cover all the cases, use the correct variable values, and so on.

There are many versions of RAND-ABEL tables, and many special features. For example (Shapiro et al., 1985a,b): (1) the number of and logical relationships among variables in a decision table can be varied; (2) automatic explanations can be produced reproducing the lines of decision tables that fire, and attaching the appropriate comments that appear in the code as footnotes to the table; (3) one can use relational operators like < and > in tables to reduce the number of lines; and (4) there exist different types of tables that provide lengthy sets of instructions in a natural format.

By contrast with software engineering techniques for using tables for pseudocode system specifications or techniques for entering rules as formatted data (see discussion in DeMarco, 1979), RAND-ABEL tables appear directly in source code--replacing long chains of If... Then... Else clauses. Properly designed rule-based programs can use tables for a large fraction of the rules, producing highly transparent, flexible, and self-documenting code.

In practice, these features of RAND-ABEL have proven extremely valuable to analysts, programmers, and reviewers, and should be seriously considered for other languages as well. In effect, the table features exploit human capability to work two-dimensionally when

studying information. In this respect, they (as well as the increasingly familiar spread-sheet programs and full-screen editors) represent a new paradigm of computer science.

4.7 The Prototype System

The prototype system for automated war gaming was demonstrated to the government late in 1984 and greatly improved over the course of the next six to nine months, with research continuing. The RSAS program currently contains the equivalent of about 120,000 lines of C-code (much of it in RAND-ABEL). An automated war game requires about 6 megabytes of storage, with current mass storage being 50 megabytes or more, and takes less than a half day (the time required depends in part on how many look-ahead projections the agents make). By and large, reactions to the prototype were highly favorable, particularly with respect to: (1) the natural representations (i.e., objects corresponding to important commands); (2) the hierarchical treatment of agents; (3) the NCL's situation assessment structure; (4) the overall emphasis on user interfaces, interactivity, and graphics; (5) the architecture allowing enrichment without change of structure; (6) the highly flexible nature of the models; (7) speed; (8) the inclusion of realistic command and control heuristic rules in preference to highly simplified optimizing algorithms; and (9) compactness (the potential exists for a single analyst to comprehend and operate the entire system).

As of late 1985, the development of decision models and Force Agent was reasonably balanced with a number of synergisms. Force Agent modellers had absorbed the expert-system paradigm and developed heuristic command and control models (in C) on the basis of experience with human play; those developing analytic war plans were efficiently calling on the Force Agent's capabilities to avoid having to reproduce clumsily, in rule-based code, the types of operational- and tactical-level decisions and orders for which algorithmically structured models are especially suited.

Although it is dangerous to make such statements, it seems that with respect to the principles of interest in this book--and in particular the feasibility of constructing large-scale complex systems combining features of both knowledge-based and simulation modelling, the most difficult conceptual and technical problems have been surmounted. What exist, however, are prototypes. The effort to enrich those models and apply them will be an exciting and challenging effort for years into the future, as will expanding the newer techniques to reach their full potential.

5. CONCLUSIONS

This final section of the paper attempts to draw lessons from the experience of a large and complex effort, lessons related to the paradigms from AI, the management challenges of developing software mixing knowledge-based and simulation techniques, and the opportunities for contributing to policy analysis.

The first conclusion should probably be that the paradigms of AI have proven enormously powerful in Rand's work. Especially significant are such concepts as nearly decomposable hierarchies, process models of behavior, and heuristic decisionmaking under conditions of complexity and uncertainty (see especially Simon, 1980, 1982; and Cyert and

March, 1963). All of these are an excellent basis for system architecture and are still insufficiently appreciated after two decades of exposition.

Another set of paradigms with origins in AI relates to transparency and comprehensibility. In the author's experience, modern computer science techniques have proven invaluable in comprehending, managing, and explaining complexity--in both software development and applications of the game-structured simulation. The effort to develop a programming language that could be read and modified by good analysts with only modest programming skills has been extremely worthwhile and has served to narrow the man-machine gap significantly, as have such techniques as postprocessor color graphics to describe simulation results (Bennett et al., 1985).

The paradigms from current expert-system work are more arguable as the basis for work of the type described here. In particular, the author is skeptical about finding generic knowledge engineers (as distinct from domain specialists collaborating with computer scientists to learn the necessary paradigms and techniques of AI) and about generic software environments as a panacea for problem solving: the limiting factor will continue to be finding individuals capable of exploring the particular problems in depth. Also, while it is important to have general concepts to guide work in knowledge-based modelling and simulation (e.g., concepts well described in Hayes-Roth, Waterman, and Lenat, 1983), it can be highly counterproductive to allow the language of abstractions and AI jargon to permeate the work place--especially if success depends upon interdisciplinary work, or even on a combination of AI conceptualizing, analytic modelling, and complex system programming. In part because of communication problems, the value of early prototypes, walkthroughs, and other mechanisms for producing something tangible early cannot be easily exaggerated. So also should modellers adopt software engineering techniques for design (e.g., DeMarco, 1979).

Another conclusion supported by work at Rand and elsewhere is that success in combining knowledge-based modelling and simulation for complex problems will require an astute combination of AI thinking and more traditional "hard" analysis: adopting heuristic approaches can be an excuse for fuzzy thinking and the failure to develop sound theories or straightforward algorithms. One suggestion is that AI specialists be used as conceptual architects, analysts (including decision analysts) used as the knowledge engineers, and system programmers be used for rigorous system design and implementation.

One conclusion from the development effort described here should be evident to readers: large and complex problems require time, money, and effort. Rand's work to develop the RSAS began in 1980, with the first prototype system not emerging until the end of 1984. This time could have been shortened somewhat, but not dramatically.

Lastly, it seems appropriate to observe that the advent of techniques combining knowledge-based modelling and traditional simulation will make it possible to exploit for the first time some of the richest ideas of the last thirty years--ideas arising from the social sciences on such issues as organizational behavior, bounded rationality, and the role of cognitive style in decisionmaking. In the past, it has been difficult to apply these ideas to practical problems in part because they did not lend themselves well to predictive modelling of the sort easily accomplished with more classical concepts such as profit maximization, game-theoretic solutions of simplified problems

ignoring uncertainty, and simulation. As a result, there continues to be a widespread tendency to approach policy-relevant modelling with the classic tools even when they are inappropriate. Now, it would seem (at least to the author) that it should be possible to do much better--primarily because it is now far easier to build rule-based models that can be large, complex, and realistic--and yet efficient and transparent. If this notion proves valid--in applications going beyond what this paper has discussed--simulation modelling will become an essential tool at the fingertips of managers and other decisionmakers (or at least their trusted staff).

REFERENCES

Bennett, Bruce W., Arthur M. Bullock, Carl M. Jones, and Paul K. Davis (1985). The RSAC Theater Warfare Model, The Rand Corporation, Santa Monica, California.

Builder, Carl (1983). Toward a Calculus of Scenarios, N-1855-DNA, The Rand Corporation, Santa Monica, California.

Carbonell, J. (1978). POLITICS: Automated Ideological Reasoning, In: Cognitive Science, 2.

Cyert, Richard M., and James G. March (1963). A Behavioral Theory of the Firm, Prentice-Hall, Inc., Englewood Cliffs.

Davis, Paul K. (1982). Concepts for Improving the Military Content of Automated War Games, P-6830, The Rand Corporation, Santa Monica, California.

Davis, Paul K., Steven C. Bankes, and James P. Kahan (1985). A Prototype Model of National Command Level Decisionmaking, R-3290-NA, The Rand Corporation, Santa Monica, California.

Davis, Paul K., and Peter J. E. Stan (1984). Concepts and Models of Escalation, R-3235, The Rand Corporation, Santa Monica, California

Davis, Paul K., Peter J. E. Stan, and Bruce W. Bennett (1983). Automated War Gaming As a Technique for Exploring Strategic Command and Control Issues, N-2044-NA, The Rand Corporation, Santa Monica, California.

Davis, Paul K., and James A. Winnefeld (1983). The Rand Strategy Assessment Center: An Overview and Interim Conclusions About Potential Utility and Development Options, R-2945-DNA, The Rand Corporation, Santa Monica, California.

DeMarco, Tom (1979). Structured Analysis and System Specification, Prentice Hall, Inc., Englewood Cliffs.

Dewar, James A., William Schwabe, and Thomas L. McNaugher (1982). Scenario Agent: A Rule-Based Model of Political Behavior for Use in Strategic Analysis, N-1781-DNA, The Rand Corporation, Santa Monica, California.

Elzas, M. S. (1984). Concepts for Model-Based Policy Construction. In: Simulation and Model-Based Methodologies: An Integrative View, T. I. Oren et al (eds). Springer-Verlag, Heidelberg.

Fain, J., F. Hayes-Roth, H. Sowizral, and D. Waterman (1982). Programming in ROSIE: An Introduction By Means of Examples, N-1646-ARPA, The Rand Corporation, Santa Monica, California.

Forrester, Jay W. (1969). Urban Dynamics, M.I.T. Press, Cambridge.

Glaser, Charles, and Paul K. Davis (1983). Treatment of Escalation in the Rand Strategy Assessment Center, N-1969-DNA. The Rand Corporation, Santa Monica, California.

Goldberg, A. and Alan Kay (1976). Smalltalk-72 Instruction Manual, Xerox PARC, SSL 76-6.

Graubard, Morlie H., and Carl H. Builder (1980). Rand's Strategic Assessment Center: An Overview of the Concept, N-1583-DNA. The Rand Corporation, Santa Monica, California. Also in: Rand's Strategic Assessment Center: An Overview of the Concept (1982), Policy Sciences.

Hall, H. Edward, Norman Z. Shapiro, and Herbert J. Shukiar (1985). Overview of RSAC System Software: A Briefing, N-2099-NA, The Rand Corporation.

Hayes-Roth, Frederick, Donald A. Waterman, and Douglas B. Lenat (Eds) (1983). Building Expert Systems, Addison-Wesley Publishing Company, Inc., Reading, MA.

Janis, Irving L., and Leon Mann (1977). Decision Making, The Free Press, New York.

Kahan, James P., William L. Schwabe, and Paul K. Davis (1985). Characterizing the Temperament of Red and Blue Agents--Models of U.S. and Soviet Decisionmakers, N-2350-NA, The Rand Corporation, Santa Monica, California.

Marshall, A. W. (1982). A Program to Improve Analytic Methods Related to Strategic Forces, In: Policy Sciences 15, Elsevier Publishing Company, Amsterdam.

McArthur, D. and Phillip Klanr (1982). The ROSS Language Manual, N-1854-AF, The Rand Corporation, Santa Monica, California.

Schank, R., and R. Abelson (1977). Scripts, Plans, Goals and Understanding, Lawrence Erlbaum Associates, Hillsdale, New Jersey.

Schwabe, William, and Lewis M. Jamison (1982) A Rule-Based Policy-Level Model of Nonsuperpower Behavior in Strategic Conflicts, R-2962-DNA, The Rand Corporation, Santa Monica, California.

Shapiro, Norman Z., H. Edward Hall, Robert H. Anderson, and Mark LaCasse (1985a). The RAND-ABEL Programming Language: History, Rationale, and Design, R-3274-NA, The Rand Corporation, Santa Monica, California.

Shapiro, Norman Z., H. Edward Hall, Robert H. Anderson, and Mark LaCasse (1985b). The RAND-ABEL Programming Language: Reference Manual, N-2367-NA, The Rand Corporation, Santa Monica, California

Simon, Herbert A. (1980). Sciences of the Artificial, M.I.T. Press, Cambridge.

Simon, Herbert A. (1982). Models of Bounded Rationality: Behavioral Economics and Business Organization, M.I.T. Press, Cambridge, 1982.

Steeb, Randall, and James Gillogly (1983). Design for an Advanced Red Agent for the Rand Strategy Assessment Center, R-2977-DNA, The Rand Corporation, Santa Monica, California.

Steinbruner, John D. (1974). The Cybernetic Theory of Decision: New Dimensions of Political Analysis, Princeton Press, Princeton.